Grid Technology Overview and Status

Geoffrey Fox[1,2], Alex Ho[2], Marlon Pierce[1]
[1] Community Grids Laboratory, Indiana University
[2] Anabas Inc.
June 7, 2005

1 Introduction

This white paper summarizes the current state of Grid technologies [Foster99A] [Foster2004A] [GGF-A] [Berman03A] [GapAnalysis] with particular attention to their possible relevance to DoD applications. In Section 2, we describe different types of Grids and contrast their use and implementation with clusters and massively parallel systems. A more extensive survey of such styles of Grids can be found in [GapAnalysis]. Section 3 summarizes the technologies divided into different classes. Section 4 describes the concept of "Grids of Grids" that is analogous to the well known "System of Systems". This concept expresses a hierarchical system model where we do not build a single monolithic system or grid but rather stitch together a system from component subsystems or subgrids that need not be architecturally homogeneous.

The review is reasonably general but there are particular comments on the work of Anabas Inc. and the Community Grids Laboratory as these highlight particular prototype opportunities. There is also special references to some existing DoD related Grid activities such as the work linking HLA and Web service technologies [XMSF1].

2 What is a Grid?

Here we try to distinguish four related networked systems

1) *Classic Massively Parallel Machine* such as the IBM SP series. These are a networked collection of nodes with a custom high performance network whose aggregate bandwidth scales proportionally to the number of nodes. The latency for small internode messages is a few microseconds. Good performance on many parallel applications requires ratio of communication times to calculation times that is not much larger than 10 to ensure low communication overheads [Dongarra02A]. The small messages are common in many cases and the low latency is needed to get good efficiency in this case.
2) *Typical cluster* which is similar to an MPP but constructed from commodity components with usually competitive node performance and bandwidth but often substantially poorer latency in the 100-1000 microsecond range.
3) *Computing Grid* is a distributed system of networked computers which can be

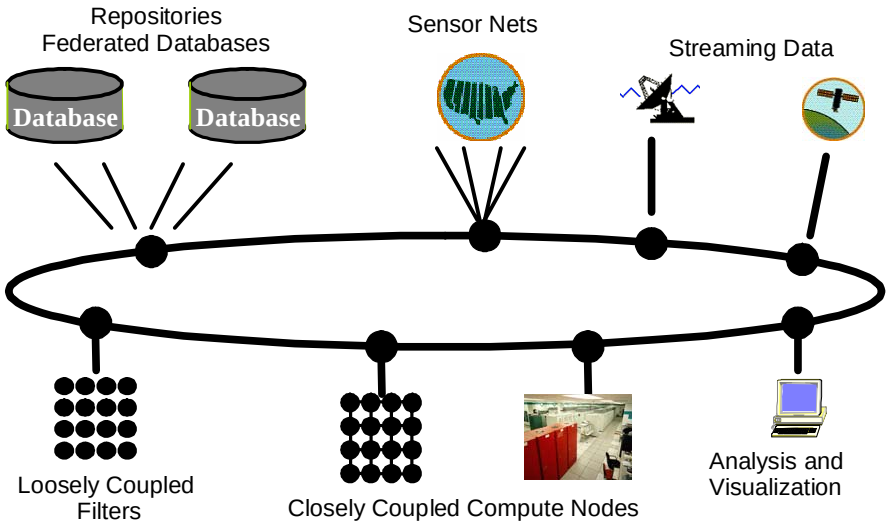

Fig. 1: Information Grid with Sensors, Satellites, databases, high performance computers, clusters and filters (independent machines)

very heterogeneous; in particular parallel machines and clusters can be nodes of a Grid. Another well-known case is the "Desktop Grid" consisting of "all the desktop machines" either in an Enterprise (the Condor model [Condor]) or in the world (the SETI@Home model [SETI]). Grids are heterogeneous in both computing nodes and networking and can have inter-node latencies of 100-1000 milliseconds as is typical of wide area networks. A geographically localized Grid could have inter-node latencies of around a millisecond. Computing Grids grant remote users the privilege to directly access computing resources.
4) *Information Grid* shown in fig. 1 is a network of computers, data repositories (both file and database) and sensors. Information grids are characterized by their use of metadata models and services to describe, organize, and provide controlled access to scientific data and resources [GapAnalysis]. Metadata is simply "data about resources" and may be used to describe a) characteristics of large permanent scientific data sets; b) ephemeral information such as computing loads on HPC systems; c) feeds for streaming data; and d) information about groups, individuals, projects, and so forth. A related problem in this area is "data

provenance" or "intellectual property" descriptions [myGrid-D]. This is used to describe who created or owns a particular piece of data, how was it created, what assumptions were made, what is the quality of the data, and so forth. Information Grids will be a focus of this paper and will be more fully described in subsequent sections. The term data grids [Venugopal05A] is often used and this sometimes refers to what we call information grids and sometimes to computing grids like that for particle physics [LCG] which were termed compute/file grids in [GapAnalysis] as they emphasized large scale computing on data placed in files rather than the streaming sensor or database model associated with information Grids.

Each of the four networked systems described above has applications for which they are suited and those for which they are less well optimized. The suitability can reflect either functionality and/or cost performance. For example, only class 1) can efficiently execute many parallel applications. On the other hand, the cost per "floating point operation" of class 2) is perhaps half that of class 1) (MPP's) and Grids can usually offer the very best cost performance of all. However this computational performance can only be realized on problems not needing the low latency synchronization and system integration only available in MPP's. This implies that Grids for example cannot easily realize the dream of meta-computing – linking multiple sites together as a single supercomputer – unless the problems can be decomposed into essentially independent tasks. There is not only the problem of very high network latencies (up to 100,000 larger than that of an MPP) but the administratively hard problem of co-scheduling – reserving large blocks of time simultaneously on geographically and administratively distinct machines. There are two important application scenarios where Computing Grids are very appropriate

a) The *unmanaged* or *managed Desktop Grid* where one has a very large set of related jobs which run independently on a pool of desktop class computers. This is familiar from "idle cycle stealing" projects like SETI@Home and in the managed case where the particle physics community expects to keep tens of thousands of computers running continuously each analyzing separate events from CERN's Large Hadron Collider LHC [Condor] [EDG-A] [LCG]. This Grid is arranged hierarchically with central, "national" and "university-level" tiers sharing the load roughly equally. The Grid components are typically each a large cluster. Note this particular application involves substantial data management problems as in full operation around the year 2010 the LHC will produce 10's of petabytes of data per year. This data will demand substantial network bandwidth but typically be staged ahead of time and have a traditional file-based computing structure as opposed to database model of Information Grids.
b) The *seamless access* capability illustrated by the Gateway project from PET [Haupt03A]. Here there is a Grid containing multiple simulation engines and one provides a portal enables submission of a given job at one of the Grid-enabled nodes. As well technology to standardize job entry, staging of files between nodes and client is typically supported in such systems. Further one provides a uniform link to machine and job status information in fashion familiar from the NPACI

HotPage resource [HotPage]. Unicore is perhaps the best known seamless access project [Unicore-A].

There is a related application category which bridges computing and information Grids.

c) *Pipelined Grid* resources are an important case as they illustrate the main reason why Information Grids are less sensitive to latency than Computing Grids. Information can typically be streamed from sensors or data repositories so that after a small start-up delay, the large WAN latency is irrelevant. The simplest case of this is a Data/Sensor/Instrument source feeding a computer which itself feeds a user analysis and visualization station. Several early Grid successes fell into this class [Laszewski02A].

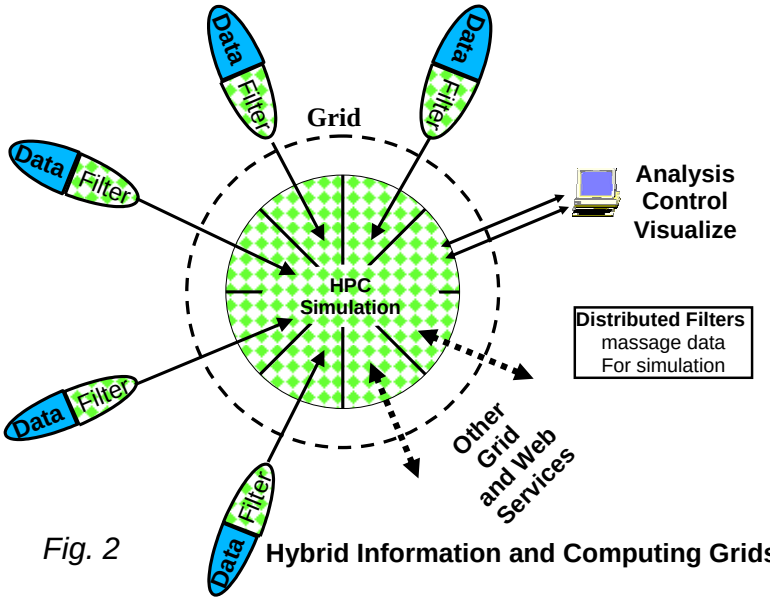

Fig. 2 **Hybrid Information and Computing Grids**

Information Grids are well illustrated by the Virtual Observatory and Bioinformatics examples

d) *Virtual Observatories* (VObs) are set up in many fields based around real-time sensors [iVOA]. The initial example comes from astronomy where for example the NVO (National Virtual Observatory) allows access to the results of many different physical observatories linking optical, radio and infrared data. This leads to a new approach to such fields stressing the integration and comparison of results from different data-gathering projects. Earth and environmental science are setting up similar VObs's combining sensor nets, satellites and data repositories as illustrated in fig. 1. A Grid is natural for such applications because not only are the original data-gathering instruments distributed but typically major telescopes each have their own specialized data archive which are also scattered around the world. As a measure of the complexity of an astronomical VObs, this academic field currently has some 10,000 users and 200 repositories worldwide.

e) Bioinformatics has spawned several Grid projects which provide access for the researcher to the growing number of databases in the field. These summarize the results of experiments of many different types and perhaps smaller in volume but

much more demanding in the curation requirements [Curation-A] to ensure databases record high quality data. EBI (European Bioinformatics Institute) [EBI] and NCBI (National Center for Biotechnology Information) [NCBI are major organizations providing many of the important databases. As well as the heterogeneous data, this field requires dynamic use of filters (such as the well known BLAST Gene sequence optimizer) which fetch data from the Grid databases and deliver results to the researcher. Virtual observatories also mix Grid computing and data access with image processing as a typical application. In this case each filtering is of the Desktop Grid class as one needs to run multiple instances of the filter on many different data selections.

f) Computational chemistry is facing many of the same requirements as bioinformatics. The traditional journal publication approach is far too slow for publishing data to the chemistry community, so distributed data base systems are being developed to house this information. Such data has many metadata requirements, such as who created it and how, where are the associated scientific articles describing the experiment or calculation, and so forth. Such metadata types are termed data provenance, or pedigree. Chemistry information grids also require community curation and annotation. Particular data sets may be "blessed" at some future time, or conversely may be labeled by other researchers as being of dubious quality. The DOE's Collaboratory for Multiscaled Chemical Science (CMCS) is an example of one such project [CMCS].

We have identified four classes of networked computing resources; classic MPP's, clusters, Computing Grids and Information Grids. All are important components in a high performance computing environment and have different application categories where they excel. MPP's and clusters are aimed at tightly coupled problems decomposed with classic parallel computing software and algorithms. These are essential parts of most Grids but the greatest opportunity for Grids is not linking several such supercomputers in real time but rather using one such simulation engine driven by a distributed collection of filters and data resources. This leads to a hybrid Grid architecture shown in fig. 2.

IBM has announced major Grid and Autonomic (robust, self healing and self-adaptive) computing initiatives addressing such problems [Horn01A].

3 Grid Technologies and Capabilities

3.1 Background

The Grid field is moving very rapidly and unlike say early Globus (GT2) systems current Grids [Foster04A] [Berman03A] are built in terms of Web services exchanging messages. The Globus Toolkit (GT2) is now replaced by the major new GT4 release [Globus-GT4]. The Grid architecture can be contrasted with collaboration systems that are built in terms of an event bus that shares state changes in distributed entities; we explore this analogy more deeply in the Collaborative Grid subsection below. Our analysis builds on our own innovative Grid architecture research [Berman03A] [WSGrids], [Aktas04A] which gives us our own leading edge technologies and a good understanding of best practice internationally. Our many activities in the Global Grid

Forum also gives us useful insights [GGF-A]. We also note our detailed report [GapAnalysis] summarizing the experiences of the UK e-Science project [UKeS-A] in using existing Grid systems such as that from the Globus team.

Several US government agencies have major Grid initiatives including the Information Power Grid from NASA [IPG] and the Science Grid from DoE [Johnston03A]. These two initial activities are no longer active but for example NIH now has several major Grid projects in the cancer [caBIG] and biomedical research [BIRN] areas. NSF is pursuing a Cyberinfrastructure initiative which embodies the Grid technical vision described here combined with the model of science and engineering comprised of collaborative interdisciplinary distributed teams [NSF03A] – the Collaboratory concept introduced by Bill Wulf over a decade ago [Wulf89]. The e-Science project in the United Kingdom has this vision with a particular emphasis on information Grids and has made substantial progress with pilot projects and aims at "production deployment" in 2006. In this spirit, they have proposed a new OMII (Open Middleware Infrastructure Institute) worldwide collaboration [OMII] to coordinate key software architectures and implementations [GapAnalysis]. Grids are being pursued actively across the world with Europe and Asia very active. China for example has several Grids with both research and commercial applications.

Grids have a layered "service architecture" where services are similar to the older distributed object model but with a looser coupling allowing greater robustness and better scaling. The difference (or not) between distributed objects and services is hotly debated

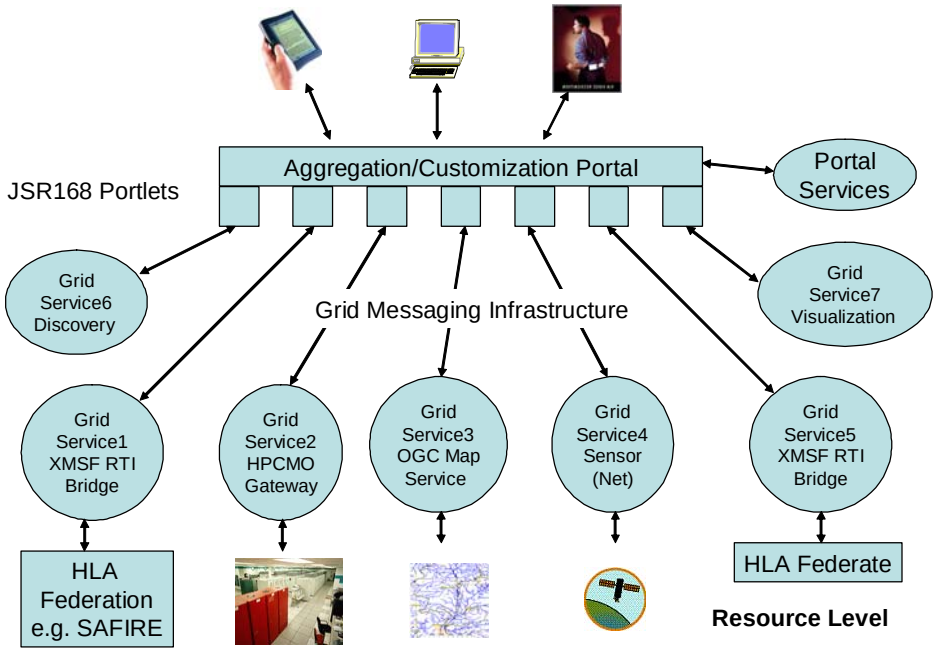

Figure 3: Grid Enhanced XMSF based Modeling and Simulation System with Clients, Portal, Grid Services and Resources including HLA/DIS subsystems

and is discussed further in several places including [WSGrids] and [Vogels03A]. The Remote Procedure Call (RPC) at the heart of Java (RMI) and CORBA is replaced by a simpler asynchronous messaging model in Grids and Web services

This section provides an overview of Grid technologies divided into service areas with a nominal DoD Grid illustrated in fig. 3.

3.2 Portal Development:

The area of Grid Portals has made excellent progress and is relatively well understood. Portals and their associated services provide the user access to the available Grid services and allow both support of seamless access discussed in [Fox05E] and the construction of problem solving environments [Haupt03A] [Fox03A]. The use of handheld devices as the user interface is part of universal access features of current portal work [Oh03A]. One has multiple Web Services with user-facing ports – each producing a fragment of a user interface. These fragments are integrated into full user interfaces by the portal containers summarized below in Table 1. There are two critical interoperability standards. The Java JSR-168 portlet specification defines how each Web service user interface fragment is defined in the container. The Web service for remote portlets specification defines the protocol by which the user-facing web service ports interact with the client or container [WSRP]. The Community Grids Lab (CGL) has developed numerous portal systems, many in collaboration with a broad Grid community [Gannon04A]. CGL's work includes the following:

a) The Gateway Portal system was developed and deployed at Wright Patterson Air Force Base (at ASC MSRC) and at Aberdeen Proving Grounds (ARL MSRC). This portal system, compliant with DOD security, supported job submission, batch script generation for specific applications (such as ANSYS), file management (uploading, downloading). See [Pierce02A] [Gateway] for more information.
b) We developed the DOD Online Knowledge Center OKC prototype system, which introduced the use of the portlet architecture in DOD environments. We also developed an XML and JMS based messaging system, which we used to build newsgroups, citation management systems, and RIB-compatible [RIB] metadata management systems [Aydin03A] [Balsoy02A].
c) The QuakeSim Portal is being developed for NASA JPL to support earthquake modeling and simulation codes. QuakeSim pioneered the use of both Portlets and Web Services for computational portal building. See [Quakesim] for project information. The portal is available from [QuakesimCGL] and described in Ref [Aktas04A].
d) CGL leads the Open Grid Computing Environments (OGCE) project [OGCE]. This NSF-funded consortium of six universities produces general purpose, reusable portal components (portlets) for computational grid portals.
e) We have developed collaboration portlets especially for audio-video conferencing [GlobalMMCS] and integrated both hand held and desktop clients into this architecture [Oh03A].

In building DoD portals, we suggest a portlet architectures as clients to remote Grid Services, as followed in our QuakeSim and OGCE projects. The portlet approach allows computing portals to be built out of reusable components. These components' displays may be customized and component access may be controlled through a portlet container.

Portlets may be reused between different containers, so an investment of time and effort to a particular vendor's product does not tie one to that vendor in the long term. Table 1 gives a list of compliant containers. Important links are [GridSphere] [Jetspeed] [uPortal]

Container Vendor	JSR 168 Compatibility	WSRP Compatibility	Stable Release	Open Source
uPortal	Yes	Yes	Yes, 2.4.1	Yes
GridSphere	Yes	No	Yes, 2.0.x	Yes
Sakai	No	Yes	Yes, 1.x	Yes
Jetspeed1	No	No	Yes, 1.5	Yes
Jetspeed2	Yes	No	No (in development)	Yes
eXoPlatform	Yes	No	No, but 1.0 Release Candidate Available	Yes
IBM WebSphere Portal	Yes	Yes	Yes, 5.0	No
Sun Java System Portal Server	Yes	Yes	Yes, 6.0	No
BEA WebLogic Portal	Yes	Yes	Yes, 8.1	No
Oracle Application Server Portal	Yes	Yes	Yes, 9	No

Table 1: Leading Portal Containers and their status and standards compliance

We can expect Grid portal and workflow technologies to be the implementation vehicle of choice for future Problem Solving Environments.

3.3 Web Service Grids Architecture

Currently there is general agreement that Grids [Foster04A] {Berman03A] should be built on top of Web Services but some significant differences and uncertainties in detailed implementation both from fundamental architectural principles and pragmatic strategy [WSGrids]. At the lowest level there are a suite of specifications commonly called the WS-* and these include areas like the basic WSDL and SOAP, Security, Reliable messaging, notification, transactions, workflow and state. There are some 60 often overlapping WS-* specifications proposed in the last few years, but so far only about 10% of these have been included in the influential WS-I [WS-I] interoperability profiles in a stark reminder of the rapid change and uncertainty in field. These specifications implicitly define a core Web Service and hence Grid architecture on which one builds higher level services. One can naturally define 3 service layers; WS-* core level [Ferguson03A] [Weerawarana05A], followed by an intermediate level of generally useful services covering areas such as data, jobs, execution and management where

critical aspects of the Grid emerge. The WS-RF Resource Framework [WSRF] from the Grid community lies in the WS-* lowest level and has been adopted by the important Globus system [Globus-A]. The third service layer consists of application specific services such as those dedicated to DoD specific applications like GridHLA. We need to choose both the architecture (selection of WS specifications) and implementations of the lowest and the intermediate level. The GGF OGSA working group is defining this intermediate level in great detail [OGSA] [OGSA-Globus]. The pragmatic WS-I+ approach [WSGrids] allows one to add new features – such as those in WSRF after their specification is complete and their value is clear to community. WS-I+ includes WS-Addressing [WSA] and WS-Eventing [WSE] (expected to become building block of WS-Notification [WSN]) which are key to WSRF. Recently the "two-core" approach to Grids has emerged [Fox05B]; this recognizes that there will inevitably be multiple some incompatible Web Service approaches and one should build higher level services in a way that is compatible with different low-level implementations. Note WS-I+ provides a Profile for best practice Grids that only uses conventional Web service specifications and this links well to goal of XMSF [XMSF1] to define HLA Web Service profiles for DoD Grids.

3.4 Workflow

Grid workflow captures "programming the Web or Grid" and encompasses a broad range of approaches with names like "Service Orchestration", "Service or Process Coordination", "Service Conversation", "Web or Grid Scripting", "Application Integration", or "Software Bus". It is an area of active research with different approaches emphasizing control flow, scheduling and/or dataflow. The workshop at GGF10 is a good summary of current best practice [workflow] with a comprehensive recent review in [Yu05A]. There is some consensus that although its use for dataflow is problematical, one should build upon BPEL [BPEL4WS] [activeBPEL] – the industry supported Web Service standard for which open source implementations are being developed [OMII]. This was based on [WSFL]. Important dataflow technologies are the Kepler [Kepler] and Triana [Triana-A] projects while Pegasus and Chimera from the Globus-Condor collaboration address the scheduling style of workflow [Pegasus] [Chimera]. Taverna from the myGrid project successfully integrates control and workflow [myGrid-B]. The HPSearch project from Indiana University supports workflow efficiently from a familiar scripting environment [HPSearch] [Gadgil04A]. An important issue for Grid integration with HLA will be melding the workflow ideas from the Grid with the federation ideas from HLA. Workflow technologies have wide applicability stretching from RTI-like systems, application code coupling and support of problem solving environments.

3.5 Notification

WSDL and SOAP are the basis of Grids built as distributed service systems. One can then build more sophisticated capabilities, such as notification, events, reliability, etc. These are defined by additional pieces of information in the SOAP XML message but these specifications need sophisticated enhancements to the Web Service software stack [Fox05A]. CGL has developed a messaging infrastructure NaradaBrokering, to manage these Web Service messages [NaradaBrokering]. The UK OMII [OMII] Grid open source software development project has endorsed our work and chosen NaradaBrokering

to implement the Web Service notification specifications WS-Eventing, Notification [WSE] [WSN] as well as the reliable messaging standards [WS-RM] [WS-Reliability]. We note that one can view WS-Eventing and WS-Notification as playing a similar role to MQ-Series [MQSeries] in Enterprise software and the Java Message Service [JMS] in distributed Java systems. WS-Eventing is similar to the core component – Base Notification – in WS-Notification and lacks the explicit topics and broker specification of the full WS-Notification standard. We expect notification to be important in DoD Grids as it supports active Data sources and robust subscription models. The same notification architecture underlies the publish-subscribe collaboration architecture pioneered by Anabas [Anabas] and the Community Grids Laboratory [GlobalMMCS].

3.6 Fault Tolerance

The area of fault tolerance for Grids and Web Services is still at an early stage with the most mature work corresponding to the WS-Reliability [WS-Reliability] and WS-ReliableMessaging [WS-RM] specifications for end-to-end messaging reliable delivery. These can be though of as implementing TCP-style retransmission protocols at the Web service message as opposed to the packet level. As mentioned already these standards are supported by NaradaBrokering and in fact using asynchronous publish-subscribe semantics is a system development paradigm that is intrinsically more fault tolerant than direct service-service messaging. Thus one can achieve many of these fault tolerance qualities by binding SOAP to a transport like JMS, MQ-Series or NaradaBrokering that supports publish-subscribe mechanisms.

More generally the concept of Autonomic Computing [Horn01A] captures a broader view of fault tolerance and there are natural mechanisms – for example heartbeat message services and replicated services that are being explored.

3.7 Management

The management of Web Services is understood to be very important and current specifications include WS-DM (Distributed Management) from OASIS and WS-Management from DMTF and Microsoft [WS-DM] [WS-Man].. A research implementation of this will be available from Indiana University using the HPSearch [HPSearch] scripting environment built on NaradaBrokering. There are also several commercial implementations of both standards. The breadth of applicability of these specifications is not yet clear with WS-Management being a broad set of protocols to allow Web Services to engage other electronic resources while WS-DM includes many detailed resource properties like lifetime. One application envisaged for these capabilities is the automatic loading of software (say Linux for a PC!) from a repository into a particular resource.

3.8 Data and Meta-data

This is a very active part of the Global Grid Forum with OGSA-DAI [OGSA-DAI] developing a powerful Grid service view of the federation of multiple disparate databases. Equally important is the meta-data area which stretches from the system level registry where UDDI [UDDI] is endorsed by WS-I+ to the application level where the Semantic Grid captures work on XML databases, lightweight protocols such as WS-

Context [WS-Context] and the Semantic Web [SemanticWeb] [SemanticGrid]. These latter technologies are particularly promising in developing ontologies allowing one to reason about the federation or linkage of services and there is already substantial interest in the HLA community in this area [Tolk04A] [Blais04A]. The Semantic Grid provides tools and architectures for annotation, search, reasoning about and access to Grid meta-data [SemanticGrid]. This includes a wide range of important capabilities from descriptions of particular services to information about the status of computers and jobs. The growing use of XML and standards based on this format will increase the importance of Semantic Grid metadata architectures.

Much DoD data comes from sensors and is tackled by a combination of database and the streaming technology described below.

3.9 Streams and High Performance Transport

GlobalMMCS has pioneered the use of a hybrid approach to messaging. SOAP in its conventional XML representation is used for all control messages while for high volume streams we use [Fox04A] high performance protocols [W3CBinaryXML] [MSBinaryXML] consistent with the SOAP Infoset [SOAPInfoset1] [SOAPInfoset2]. In particular we are able to support a simple low overhead enhancement of RTP to transport all audio/video streams. NaradaBrokering queues messages on all messaging links if necessary (to throttle back stream) or requested (to archive stream for replay in collaboration systems). Message queuing uses a combination of in-memory buffers, MySQL databases and flat files. Robustness of the in-memory queues is addressed by the disk storage and replication on different brokers. Robustness of disk storage is addressed using conventional methods (RAID and distributed replication) for this medium. NaradaBrokering is currently being re-engineered [Fox05A] so that it can be deployed transparently as handlers for Web Service infrastructure like Axis [Axis]. By using NaradaBrokering or similar technology as a transport and representation handler one can move between the binary and classic angle-bracketed representations of SOAP messages without content loss. Efficient binary representations of XML Infosets have been developed including SOAP Message Transmission Optimization Mechanism (MTOM) [MTOM] and XML-binary Optimized Packing (XOP) [XOP]. We are developing schemes which allow two endpoints to first negotiate the best-available transport and then proceed to use it for transfers. To accommodate legacy systems that do not use the XML format, the Data Format Description Language (DFDL) [DFDL] is an XML-based language that describes the structure of binary and character-encoded files and data streams so that their format, structure, and metadata can be exposed. This can also be used in tandem while transferring binary data using SOAP. The NaradaBrokering substrate incorporates support for several transport protocols (including parallel TCP) that can be leveraged to provide high performance transfers of such large binary data. Since the substrate allows new transport protocols to be plugged in rather easily, support for newer transport schemes can easily be incorporated. Our work here will provide support for both high-performance, high volume data transport as well as data upload and delivery for Web-enabled devices. The substrate will also incorporate a caching scheme which would be suitable for supporting high-performance distributed (HLA) simulations.

The caching scheme will be used in reducing database access times while retrieving simulation events that will be delivered reliably to participating entities.

There has already been interesting work in the DoD community on high performance XML [XSBC] with an architecture similar to ours developed by Pullen [Moen03A] with the XOM system developed at the C3I center at George Mason University [XOM] and an approach based on BEEP used by Web Enabled RTI from SAIC [XMSF3] [Pullen04A] [Morse04A].

3.10 Security

Security is particularly important for heterogeneous distributed systems and essential for e-commerce and of course DoD applications of Grid (Web Service) technologies. Grids require an extension of the traditional transport level security systems (such as SSL). Transport level security insures safe transmission of messages between two set endpoints. Grids on the other hand may need to pass messages through several intermediate hosts and may need to send messages to more than one end point. Grids thus require message-level security in addition to simple, point to point transport level security mechanisms. Current Public Key and Kerberos capabilities for authentication and authorization may be implemented in a message-based Web Service security model whose message-based model has advantages over previous connection-based schemes. This integration of Grid and Web Service security is still a "work in progress" and our approach should be fully compatible with any approaches that emerge. WS-Security [WS-Security] is expected to be the overall framework with WS-SecureConversation [WS-SC] used for streaming. We expect progress in areas like linking the Globus delegation model with Shibboleth [Shibboleth] and Web Services with key problems being fine grained authorization and support of trust across multiple services (workflow) and for the long time periods that a Grid application might last. NaradaBrokering has a security model [Pallickara03A] compatible with this emerging Web service model

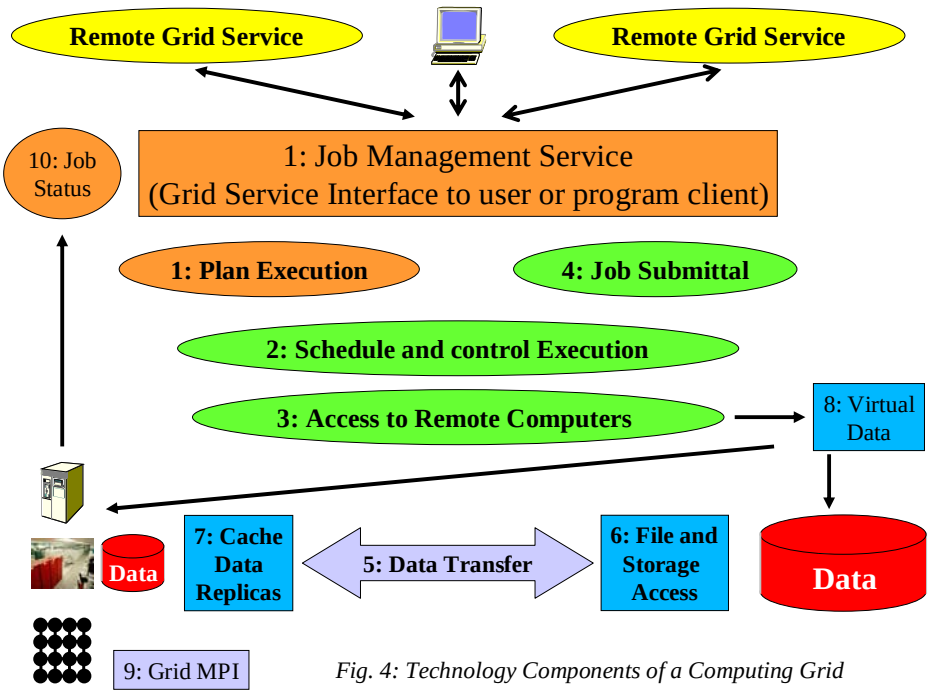

Fig. 4: Technology Components of a Computing Grid

3.11 Computing Services

There are a suite of important services associated with the myriad of activities associated with running a job. These are sketched in fig. 4, and are comprised of the many services needed to support distributed computing models. As well as scheduling, planning and job submission familiar from Condor [Condor] and Globus [Globus-A], one needs caching and file management combined of course with the three services described above. File services include Grid flavors of shared file systems ("GridNFS") and of data transport (GridFTP).

3.12 Network Grid Services

Network Services including monitoring, reservation and routing have not received so much attention but should become in future Grids as we need high performance deployment respecting security, resilience and reliability issues. NaradaBrokering integrates network performance information into its Grid messaging routing algorithms and the GGF has a working group NM-WG setting XML standards for network performance data [GGF-A]. The field is reviewed in more detail in [GapAnalysis] but is not discussed further here as there is little Grid specific work in the field.

3.13 Collaborative Grids

Collaboration or the sharing of Web services unifies areas such as Access Grid [AccessGrid] and Peer-to-peer networks. Rapid progress is being made and we can again expect greater security and robustness to result for all collaboration tools. Grids, e-Science and CyberInfrastructure are often discussed in terms of virtual organizations (VO) where asynchronous and synchronous collaboration are essential to support VO's. An integration of Grid, peer-to-peer, Web service and collaboration environments was explained in chapter 18 of [Berman03A]. A key idea is that one can make Web (and hence Grid) services collaborative rather straightforwardly as their state can only be changed by input messages and the current state is only defined by output messages. This led to the concept of shared input-port and shared output-port collaborative services.

Anabas originally developed some of the core ideas in this area building the first collaboration environment [Anabas] supported by general purpose publish/subscribe infrastructure – initially the Java Message Service [JMS]. It was found that this introduced only a few millisecond overhead that was negligible and allowed a much more general flexible approach to collaboration than that in well known commercial products like Webex and Placeware or the open source VNC [Groove] [JXTA] [Centra] [Placeware] [WebEx] [Interwise] [VNC]. State change events are published by the "master" client to a topic and collaborating clients subscribe. Later CGL designed a sophisticated messaging environment, NaradaBrokering [NaradaBrokering], that offered key new capabilities including support for JMS and JXTA programming APIs, compatibility with Web services, fault tolerance and support of multiple protocols including both TCP, parallel TCP and UDP in the same publish-subscribe framework. Anabas is now using NaradaBrokering in its application specific products. The UDP transport is used in the GlobalMMCS project [GlobalMMCS] which builds a service

oriented audio-video conferencing system around NaradaBrokering. GlobalMMCS supports both Access Grid and VRVS clients [AccessGrid] [VRVS]. CGL is studying integration of GlobalMMCS with CEE from AFRL [McQuay04] [CEE00] and KnowledgeKinetics [KK] in a project funded through Ball Aerospace. The use of CGL XGSP technology (XML General Session Protocol) to support dynamic collaborative environment with multiple roles is very promising [XGSP].

4 Grids of Grids

In [GofG], we introduced the concept of Grids of Grids of Simple Services". Here we review and extend this and show how it can address several issues of importance to DoD including the "right sizing of services", "an architecture for dealing with legacy systems" and a "strategy for modularizing the design and implementation of systems (grids)".

Consider any (software) problem you like and imagine how it would look in a traditional approach of a decade or so ago. One would get monolithic chunks of software in some language like C++ or ADA. This would be divided into methods or subroutines and we would be instructed to build it in modular fashions using libraries and well defined interfaces. As technologies developed we added new languages like Java and better software engineering processes which still however focused on modularity within a chunk of software divided into objects and/or components. For example you will find this software structure if you inspect well known open source projects such as Linux or the Java at the Apache site [Apache] . One can convert such code into services by specifying each of interfaces in XML and providing a Web Service wrapper. This activity is important for jump starting our collection of services but I would view it as only appropriate for legacy systems and a less than optimal approach to new software systems. For example looking at the many different Apache projects, one will find many related but different implementations of common subservices like security, file access and user profile. Building a system combining several projects would often require an integrated approach to common services like security. This would be relatively easy if the implementation of each subservice like security was a separate Grid service with well defined message-based interfaces. However with traditional approach, the typical subservice can have an external message-based interface but unfortunately in addition many internal method linkages to other parts of the software chunk where typically it is hard to serialize the arguments. Thus subservices like security cannot be extracted from the glob and it is very hard to use components such traditional software systems even if they run excellently with service interfaces.

The above discussion allows us to identify a strategy for defining what we term simple services. Start by examining the different capabilities of one's systems. Services are distributed components that have distinct functionality – especially functionality that is usefully shared among different uses. Services must be able to achieve acceptable performance when implemented with message based interfaces and distributed platforms. There is an inevitable difference in overhead between message and method based interactions; messages could experience 100's of milliseconds in network latency while the internal method calls have a fraction of a millisecond overhead. We define simple services as those that are as small as possible given the performance implications from

the decomposition. Such simple services are then the unit for which one uses traditional programming models and languages. This is the proposed strategy for "right-sizing" services.

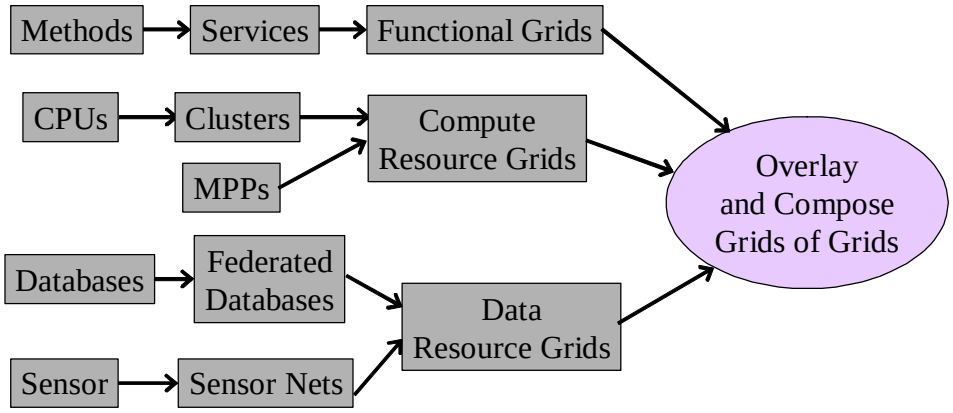

Fig. 5: Composing Functionality and Resources in the Grid of Grids

In fig. 5, we suggest a packaging and coupling approach that generalizes and distributes that familiar from the traditional software hierarchy:
lines of code → methods (subroutines) → objects (programs) → packages (libraries).
A single simple service is the smallest grid but we can integrate like simple services into library grids. These sub-grids are then composed into a complete "Grid of Grids" implementing the full system. Fig. 5 shows how database, sensors and compute nodes (abstracted as simple services representing "simple resources") can be federated, networked and clustered into larger units.

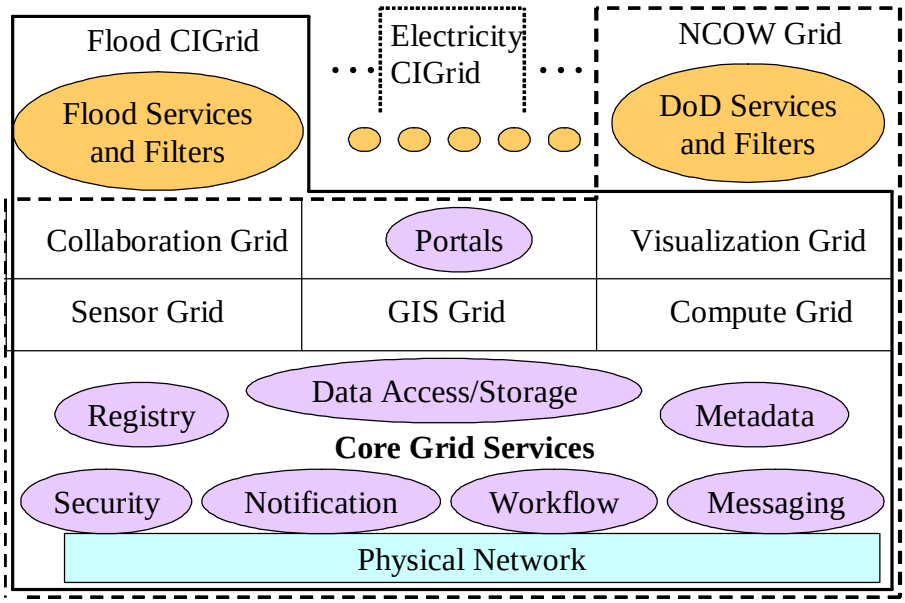

Fig. 6: Critical Infrastructure (CI) Grids built in composite fashion and linked to an NCOW (GiG) Grid

As a particular example of a Grid of Grids, Fig. 6 illustrates how one can share component Grids between critical infrastructure applications and DoD's NCOW [Fox05D]. The Department of Homeland Security has identified critical infrastructures that include Agriculture and Food, Water, Health, Industrial and Defense Base, Telecommunications, Energy, Transportation, Banking and Finance, Chemical Industry and Hazardous Materials, Postal and Shipping. The critical atomic Grids in this case include those for sensors, GIS, visualization, computing and collaboration. We also need of course the core Grid shown at the bottom of the figure with services like security, notification and meta-data. These atomic Grids can be re-used as shown in figure 6 in all critical infrastructure Grids and illustrate the important interoperability principles with which Grids are built. These CI(Critical Infrastructure) Grids are in turn customized, composed and overlaid with other Grids (such as weather, census data) for different CI communities. This way one generates Grids aimed at Public Health, Emergency Response (Command and Control) or Crisis Grids, Infrastructure Planning, Education (schools) and Training (of managers and first responders). Clearly the Grid of Grids concept can be applied recursively and dynamically.

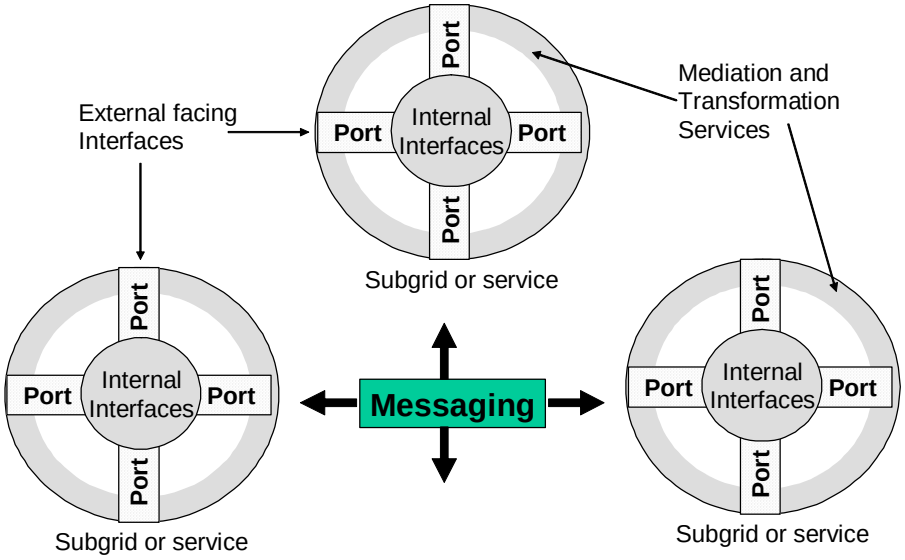

Fig. 7: Mediation and Transformation in a Grid of Grids and Simple Services

Note that both simple services and grids interact with the outside environment through messages and these messages are the only way to both impact and learn about the service or grid. For Web service based grids these messages are defined by the WSDL – where for grids this WSDL is the concatenation of the WSDL of all its external interfaces. Note that in this view all that counts are "outward facing" interfaces. Internal interfaces need not be specified to use a given grid. In particular these internal interfaces could use different flavors of Web service specification or totally different technology – methods, Java RMI, CORBA etc. Examples of different Web service flavors are WSRF or WS-I+ based systems [WSRF] [WSGrids] or more simply the two flavors of reliable messaging in [WS-Reliability] and [WS-RM]. If we assume that we use a message-oriented-middleware (MOM) implementation then all messages entering a particular simple service or grid is explicitly handled and can be transformed to confirm to the internal conventions of this grid as illustrated in fig. 7. This gives us a clear strategy for legacy systems – one identifies their outward facing Grid interfaces, defines in WSDL and builds a set of transformations that map between the system-wide Grid standards and those used internally. The same idea can be used to build virtual private grids generalizing VPN's to grid systems [Fox04B] and so ensuring particular security policies within a given subgrid. More generally support of hierarchically constructed grids of heterogeneous components gives a robust software engineering strategy with a modular software model.